DL4G - Questionnaire

Deep Learning for Games

Maurin D. Thalmann

19. Januar 2020

Dieser Questionnaire wurde basierend auf einer Card2Brain Sammlung erstellt:
Card2Brain - DL4G (Credits: Cyrille Ulmi)

Inhaltsverzeichnis

1 Sequenzielle Spiele

1.1 Was sind die Eigenschaften von endlichen-sequenziellen Spielen?

- Eine endliche Anzahl Spieler mit einer endlichen Anzahl Aktionen
- Die Aktionen werden sequenziell ausgewählt
- Es wird eine endliche Anzahl Runden gespielt
- Spätere Spieler sehen die Aktionen vorheriger Spieler

1.2 War wird unter Perfect Recall verstanden?

Perfekte Erinnerung an alle vorherigen Züge

1.3 Was ist eine Strategie?

Sagt einem Spieler, welche Aktion im aktuellen Zug auszuführen ist

1.4 Was ist ein Strategie-Profil?

Die ausgewählte Strategie eines Spielers

1.5 Was ist eine Utility- oder Payoff-Function?

Sie berechnet das Resultat für jede Aktion

1.6 Was sind die Komplexitätsfaktoren bei einer Spielanalyse?

- Anzahl Spieler
- Grösse des Suchraums (Anzahl gespielte Züge & Anzahl mögliche Aktionen)
- Kompetitiv vs. Kooperativ
- Stochastische Spiele (mit Zufall) vs. Deterministisch
- Perfekte vs. imperfekte Information

1.7 Was ist imperfekte Information?

- Das Spiel konnte nur teilweise beobachtet werden
- Man kennt bspw. nicht die Karten der anderen Spieler

1.8 Beispiele von Spielen mit perfekten / imperfekten Informationen?

Perfekt (Schach) und imperfekt (Jass, Poker)

1.9 Was ist der Suchraum?

Anzahl gültige Brettpositionen und die untere Grenze des Suchbaums

1.10 Was ist ein Suchbaum?

- Knoten sind Spielpositionen / Spielzustände
- Kanten sind Aktionen / Spielzüge
- Blätter werden durch Payoff-Funktionen definiert

1.11 Wie funktioniert Backward Induction?

- Den Baum von unten nach oben durcharbeiten (bzw. von rechts nach links)
- Immer den besten Weg für den aktuellen Spieler markieren
- Geeignet für sequenzielle endliche Spiele mit perfekter Information

1.12 Was bedeutet Rationalität?

Dass der Spieler nicht die schlechtere Alternative wählt

1.13 Welche Arten von Lösungen werden bei endlich-sequenziellen Spielen unterschieden?

- Ultra-schwache Lösung
  - Bestimmt, ob der erste Spieler einen Vorteil aus der Initialposition hat, ohne die genaue Strategie zu kennen
  - Setzt perfektes Spielen des Gegners voraus
  - Beispielsweise durch Existenzbeweise in der Mathematik
- Schwache Lösung
  - Kann ein komplettes Spiel mit perfekten Zügen aus der Initialposition durchspielen
  - Geht von einem perfekten Spiel des Gegners aus
- Starke Lösung
  - Kann aus jeder Position heraus perfekte Züge spielen
  - Kann auch gewinnen, wenn vorherige Spieler einen Fehler gemacht haben

1.14 Was versteht man unter einem Zero-Sum Game (Nullsummenspiel)?

- Der Vorteil für einen Spieler ist zum Nachteil des anderen Spielers
- Die Punktesumme für zwei Strategien ist immer gleich Null

1.15 Was sind Charakteristiken des Minimax-Algorithmus?

- Gilt nur für ein Nullsummenspiel
- Zwei Möglichkeiten / Ziele
  - den eigenen Gewinn maximieren
  - den Gewinn des Gegners minimieren

1.16 Wie funktioniert der Minimax-Algorithmus?

- Wenn der Knoten mir gehört: Aktion wählen, die den Payoff maximiert
- Wenn der Knoten dem Gegner gehört: Aktion wählen, die den Payoff minimiert
- Wenn es ein Endknoten ist: den Payoff berechnen

1.17 Was versteht man unter Search Tree Pruning?

Nicht relevante Teilbäume können weggelassen werden, reduziert den Rechenaufwand

1.18 Was sind die Regeln von Alpha-Beta Pruning?

- α ist der grösste Wert alles MAX Vorfahren eines MIN Knoten
- β ist der kleinste Wert alles MIN Vorfahren eines MAX Knoten
- Den Teilbaum abschneiden, falls er grösser als α oder kleiner als β ist

1.19 Was ist der Vorteil von Alpha-Beta Pruning?

- b = Anzahl Kanter der Knoten und m = Tiefe des Baums
- Ordnung verbessert sich von $O(b^m)$ nach $O(b^{m/2})$, halbiert also die Tiefe der Suchbäume

2 Monte Carlo Tree Search

2.1 Wieso werden Random Walks eingesetzt? (Tree Search)

- Der Suchraum ist oft zu gross für eine vollständige Suche
- Die Idee, verglichen zu Minimax, ist, bei einer bestimmten Tiefe zu stoppen und zu raten

2.2 Was ist die Idee hinter Monte Carlo Tree Search?

- Macht einen Random Walk und spielt zufällige Simulationen
- Versucht, in einer fixen Zeit möglichst viel des Suchraums zu entdecken
- Am Schluss wird der vielversprechendste Spielzug ausgewählt

2.3 Welche 4 Phasen gibt es bei Monte Carlo Tree Search?

1. **Selection**
   - Starte beim Wurzelknoten `R` und wähle fortlaufend Kinderknoten
   - Stoppe, wenn du einen Knoten erreichst, der noch nicht komplett erweitert/erforscht wurde
   - Benötigt ein Kriterum für die Auswahl der Kinderknoten, sogennante *tree policy*
2. **Expansion**
   - Wenn das Zeitlimit `L` das Spiel beendet, gib die Payoffs zurück
   - Sonst, wähle eine unerforschte Aktion und kreiere einen Knoten `C` für diese
3. **Simulation**
   - Simuliere ein Weiterspielen von Knoten `C` aus, mithilfe einer *default policy*
   - Im simpelsten Fall, spiele einfach bis zu irgendeinem Ende mit zufälligen Zügen
4. **Backpropagation**
   - Aktualisiere die gespeicherten Informationen in jedem Knoten von `C` zurück bis zu `R`
   - MCTS erwartet einen Payoff in [0,1]

2.4 Welche zwei Ansätze gibt es beim Auswählen eines neuen Knotens?

- Exploitation
  - Immer den besten Payoff wählen
  - Anhand von Beobachtungen auf der besten Maschine spielen, um Gewinn zu maximieren
- Exploration
  - Etwas Neues wählen, versuchen möglichst viel zu erkunden
  - Alle Maschinen spielen, um möglichst viel Informationen zu gewinnen

2.5 Was ist die Idee hinter UCB1 (Upper Confidence Bound)?

- Die beste Strategie ist eine Mischung aus Exploitation und Exploration
- Ergibt ein statistisches Konfidenzintervall für jede Option
- Parameter c kontrolliert den Trade-Off zwischen Exploitation und Exploration

2.6 Was ist sehr wichtig bei der Anwendung von UCB1?

Immer die Vektor-Komponente des aktuellen Spielers für die Berechnung verwenden.

2.7 Was passiert bei MCTS, wenn die Zeit abgelaufen ist?

Spielt die Aktion mit der höchsten Anzahl an Besuchen.

2.8 Was sind Unterschiede zwischen Minimax und MCTS?

- Beide Algorithmen setzen perfekte Informationen voraus
- Minimax ist nur anwendbar auf Nullsummenspiele mit zwei Spielern
- MCTS funktioniert für jedes Spiel mit perfekter Information
- Minimax optimiert Payoffs, MCTS optimiert einen Exploitation-Exploration Trade-Off
- MCTS ist ein Anytime-Algorithmus, Minimax nicht
- Monte Carlo Bäume sind asymmetrisch, Minimax Bäume sind symmetrisch

2.9 Was ist ein Anytime-Algorithmus?

Er kann eine gültige Lösung zurückgeben, auch wenn die Ausführung vorzeitig abgebrochen wird. Es wird erwartet, dass er eine immer bessere Lösung findet, je länger er ausgeführt wird.

2.10 Wie sehen die Payoffs bei MCTS aus und was wird maximiert?

- Für ein Beispiel mit 2 Spielern nimmt der Payoff-Vektor die Form $(W, N - W)$ an
- Spieler 1 maximiert W, Spieler 2 maximiert $N - W$ (implizit minimiert Spieler 2 so auch $-W$)

3 Information Sets

3.1 Was ist ein Information Set?

- Ein Information Set ist eine Menge von Knoten des gleichen Spielers
- Der Spieler kennt den vorherigen Zug nicht
- Alle Knoten müssen die gleichen Optionen bieten
- Sind immer aus der Sicht eines Spielers

3.2 Was ist der Unterschied zwischen perfekter und imperfekter Information?

- Unterschiedliche Strategien werden gewählt
- Bei perfekter Information hat jeder Knoten exakt eine Option

3.3 Was ist die Idee hinter Determinization?

- Unbekannte Karten zufällig auf die Gegner verteilen
- Danach das Anwenden der Regeln von perfekter Information (bspw. mit MCTS)
- Ergibt dann mehrere mögliche Suchbäume
- Schlussendlich über alle Bäume die Option wählen, die am meisten besucht wurde
- Manchmal werden Entscheidungen getroffen, welche gar nie eintreten können, was zu falschen Entscheidungen führen kann

3.4 Was muss bei einer Implementation von MCTS mit Information Sets beachtet werden?

- Der Baum besteht aus Information Sets und nicht mehr aus Zuständen
- Knoten entsprechen Information Sets aus der Sicht des Wurzelspielers
- Karten werden zufällig verteilt und ungültige Varianten werden ausgeblendet
- Kanten entsprechen einer Aktion, welche in mindestens einem Zustand möglich ist
- Funktioniert danach wie ein Spiel mit perfekter Information

3.5 Wie muss die UCB1 angepasst werden für Information Sets?

- N_p = Anzahl Besuche des Vorgänger-Knotens und wie oft Knoten i verfügbar war
- Knotenliste ergänzen um „wie viel mal war jede Option verfügbar“